

SQL Flow Control and Variables (PostgreSQL)

PostgreSQL supports the use of variables and flow control statements within stored procedures, functions, or anonymous code blocks (using Do blocks).

You can execute a group of SQL statements (including flow control and variables) without creating a stored procedure by using a Do block. However, if you want to call your code from outside SQL (for example, from a client application), then you should place it inside a stored procedure or function. \$ (double quoting) is a PostgreSQL substitute for single quotes to avoid quoting issues inside the Begin...End block. This means that you can use single quotes around strings without any issues.

Variables

Variables are used to hold and manipulate values in memory. They must be declared before they are used, allowing PostgreSQL to allocate storage efficiently.

Syntax

Declare

```
variablename datatype;  
variablename datatype;  
variablename datatype;
```

All variables must be declared at the beginning of the block, before any executable statements.

Assigning Values to Variables

Variables can be assigned literal values or values from queries.

Assigning a Literal Value

```
variable_name := literal_value;
```

Example:

```
Do $$  
Declare  
    v_Student_Count integer;  
Begin  
    v_Student_Count:= 100;  
    Raise Notice 'Number of students: %', v_Student_Count;  
End $$;
```

Raise Notice in PostgreSQL sends a message to the client or message window. It is similar to Print in SQL Server but should mainly be used for testing or debugging, not for returning data to the user.

Assigning a Value from a Query

```
SELECT    column_name
INTO      variable_name
FROM      table_name
WHERE     condition;
```

Example:

Do \$\$

Declare

 V_Total_Pay Decimal(8,2);

Begin

 Select Sum(Amount)

 Into V_Total_Pay

 From Payment;

 Raise Notice 'Total Payments: %', V_Total_Pay;

End \$\$;

Flow Control

You can control the execution of SQL statements using flow control structures such as If, Elself (optional), and Else (optional). You can have as many Elself statements as you want, but only one Else statement is allowed. You can use Begin and End for readability, but it is not required in If statements.

Syntax:

```
If condition Then
    Begin
        -- SQL statements
    End
Elself another_condition Then
    Begin
        -- SQL statements
    End
Else
    Begin
        -- SQL statements
    End
End If;
```

Example:

```
Do $$
Declare
    v_Avg_Mark decimal(5,2) := 75;
Begin
    If v_Avg_Mark >= 80 Then
        Raise Notice 'Excellent!';
    Elself v_Avg_Mark >= 50 Then
        Raise Notice 'Pass';
    Else
        Raise Notice 'Fail';
    End If;
```

End \$\$;

Using If Exists / If Not Exists

A common use of conditional logic is to check if a record exists in a table before performing an action. The “If Exists” clause returns True if the subquery returns at least one record, and False otherwise. Conversely, using “If Not Exists” returns False if the subquery returns at least one record, and True otherwise.

Syntax:

If Exists (Select 1 From ***table_name*** Where ***condition***) Then

-- SQL statements

End If;

Example:

Do \$\$

Begin

```
    If Exists      (Select      1
                    From        Student
                    Where      StudentID = 123) Then
```

```
        Raise Notice 'Student already exists.';
```

```
    Else
```

```
        Raise Notice 'Student already exists.';
```

```
    End If;
```

End \$\$;

Change 123 to 199899200 to see if you get a different message.

Anonymous Code Blocks (Do)

Each Do block or stored procedure executes as a single unit.

You can separate logical sections of a script simply by executing one block at a time.

Hypothetical Example (Client table does not exist in IQSchool):

```
Do $$
```

```
Begin
```

```
    Alter Table Client
```

```
        Add Column
```

```
            Active Char(1);
```

```
End $$;
```

```
Do $$
```

```
Begin
```

```
    Alter Table Client
```

```
        Add Constraint CK_Client_Active_TF
```

```
            Check (Active In ('T', 'F'));
```

```
End $$;
```

In this example, each Do block executes separately. This ensures the column is added before the constraint is created.

Notes

- Do blocks are executed immediately and are **not stored** in the database.
- Use Create Procedure or Create Function if you need to store reusable code in the database.